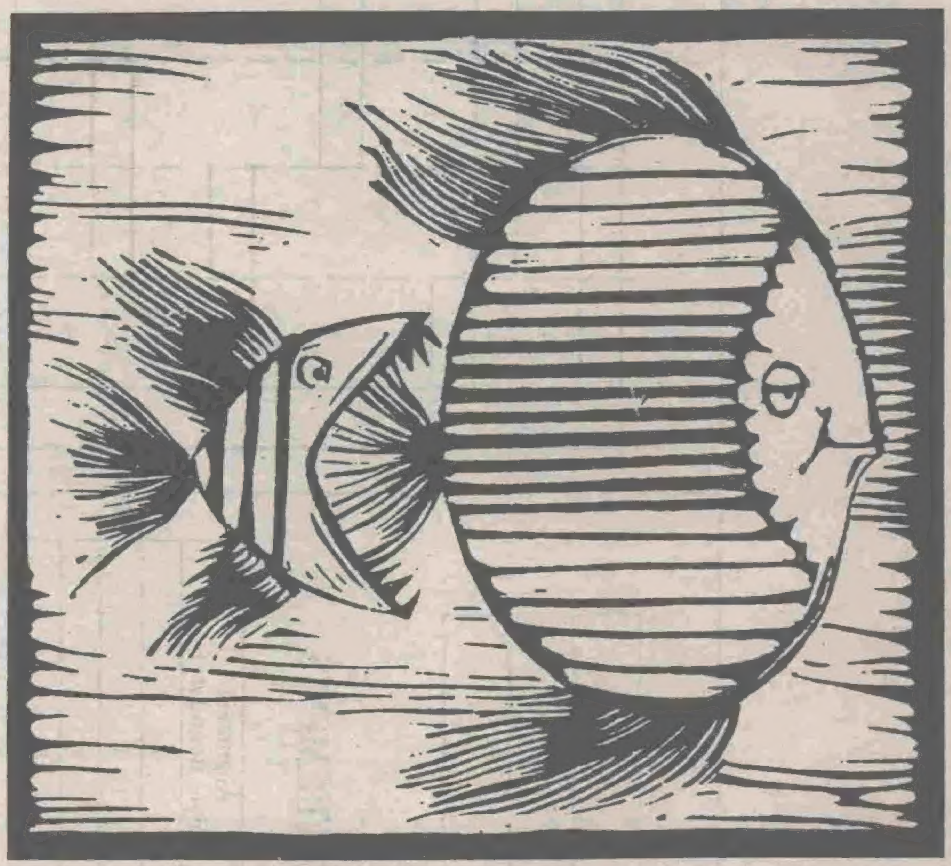
บุญเลิศ เอี่ยมทัศนา

ข้อมูลแบบโปรแกรมย่อยในเทอร์โบปาสคาล

เทคนิคการเขียนโปรแกรมย่อย
กำหนดตัวแปรเป็นอะเรย์
การคำนวณของโปรแกรมย่อย
ให้ค่าของโปรแกรมแก่ตัวแปร



ในภาษาปาสคาลให้เรากำหนดแบบข้อมูลเป็นอะเรย์ เรคอร์ดและอื่น ๆ ได้พิเศษสำหรับเทอร์โบปาสคาลคือให้เราสามารถกำหนดแบบข้อมูลเป็นโปรแกรมย่อย (Procedure Type) ทั้งในแบบที่เป็นโพรซีเดอร์และฟังก์ชันได้ ซึ่งเมื่อกำหนดแล้วสามารถกำหนดตัวแปรแบบนี้ เพื่อการให้ค่าและผ่านค่าเป็นโปรแกรมย่อยตามแบบที่กำหนดได้ในลักษณะเดียวกันกับข้อมูลแบบอื่น ๆ กล่าวคือ

การให้ค่าแก่ตัวแปร

ตามรูปที่ 1 ได้กำหนดแบบข้อมูลโปรแกรมย่อยไว้ 2 แบบคือ

- Proc เป็นแบบโพรซีเดอร์ที่ไม่มีอาร์กิวเมนต์

- CompareFunct เป็นแบบฟังก์ชันที่มีอาร์กิวเมนต์เป็นเลขจำนวนเต็ม 2 จำนวน และตัวมันเองเป็นแบบบูลีน

เมื่อกำหนดแบบข้อมูลแล้วจะกำหนดตัวแปรได้ดังที่แสดงในรูป (อนึ่งคำว่า Proc และ Funct ที่ผมนำมาใช้เป็นชื่อและกำกับชื่อ ไม่ได้มีข้อกำหนดใด เขียนไว้เป็นที่สังเกตเท่านั้น สำหรับโปรแกรมย่อยที่จะให้ค่าหรือผ่านค่าให้แก่ตัวแปรแบบนี้ได้นั้น ได้แสดงเป็นตัวอย่างไว้ในรูปที่ 1 เช่นกัน ตามข้อกำหนดว่า

1. ต้องมีส่วนหัวแบบเดียวกับที่กำหนดไว้ในแบบข้อมูล

2. จะต้องคอมพายล์ในแบบ Far Call คือที่เห็นใช้คอมพายเลอร์ไดเรคทีฟ F

3. จะต้องไม่เป็นโพรซีเดอร์หรือฟังก์ชันมาตรฐาน (คือที่มีอยู่ในเทอร์โบปาสคาลเดิม)

4. จะต้องไม่เป็นโปรแกรมย่อยในโปรแกรมย่อย

5. จะต้องไม่เป็นโปรแกรมย่อยในแบบ Inline หรือ Interrupt

หลังจากกำหนดแบบข้อมูล ตัวแปร และโปรแกรมย่อยแล้ว สามารถนำมาใช้ในโปรแกรมหลักได้ดังรูปที่ 2 โดยที่ในโปรแกรมหลักได้ให้ดำเนินการดังนี้

- ให้ค่าโปรแกรม MyOne แก่ตัวแปร oneProc และโปรแกรม My-

รูปที่ 1 แบบข้อมูลและตัวแปรแบบโปรแกรมย่อย กับโปรแกรมย่อยที่จะให้ค่าแก่ตัวแปรได้

```
PROGRAM  DemoProcedureType;
USES     crt;
TYPE     Proc = Procedure;
         CompareFunct = Function(a,b : Integer) : Boolean;
VAR      oneProc,twoProc : Proc;
         compFunct : CompareFunct;
         a,b : Integer;


{$F+}
PROCEDURE  MyOne;
BEGIN
  WriteLn('This is ONE');
END;

PROCEDURE  MyTwo;
BEGIN
  WriteLn('This is TWO');
END;


FUNCTION  GreaterThan(a,b : Integer) : Boolean;
BEGIN
  GreaterThan := a > b;
  Write(a,' is greater than ',b,' is ');
END;


FUNCTION  Equal(a,b : Integer) : Boolean;
BEGIN
  Equal := a = b;
  Write(a,' is equal to ',b,' is ');
END;


FUNCTION  LesserThan(a,b : Integer) : Boolean;
BEGIN
  LesserThan := a < b;
  Write(a,' is lesser than ',b,' is ');
END;
{$F-}
```

รูปที่ 2 โปรแกรมหลักแสดงการใช้งานตัวแปรแบบโปรแกรมย่อย

```
BEGIN
  oneProc := MyOne;  twoProc := MyTwo;
  oneProc;  twoProc;
  oneProc := twoProc;  oneProc;

  compFunct := GreaterThan;  a := 10;  b := 5;
  WriteLn(compFunct(a,b));

END.
```

Two แก่ตัวแปร twoProc ทำให้ตัวแปรแทนโปรแกรมดังกล่าว ดังนั้นเมื่อเรียกใช้ oneProc จะเป็นการเรียกใช้ MyOne เช่นเดียวกับ twoProc ดังผลการรันในรูปที่ 3 บรรทัดที่ 1 และ 2

- ให้ค่าใน twoProc แก่ oneProc ดังนั้นเมื่อเรียกใช้ oneProc จึงเป็นการเรียกใช้ MyTwo ดังผลการรันบรรทัดที่ 3

- ให้ค่าฟังก์ชัน GreaterThan แก่ตัวแปรโปรแกรมย่อย compFunct และให้ค่าแก่ตัวแปรเลขจำนวนเต็ม a และ b ซึ่งในฟังก์ชัน GreaterThan จะเทียบค่า a กับ b ให้ผลเป็น TRUE หรือ FALSE จึงนำมาใช้กับคำสั่ง

WriteLn ได้ ผลการรันจึงออกมาดังบรรทัดที่ 4

อะเรย์ของโปรแกรมย่อย

เช่นเดียวกับข้อมูลแบบอื่น ๆ เราสามารถทำเป็นอะเรย์ของข้อมูลแบบโปรแกรมย่อยได้ ดังการกำหนดแบบข้อมูลและตัวแปรในรูปที่ 4 ให้เป็นอะเรย์ขนาด 3 ช่องของข้อมูลแบบ CompareFunct และนำไปใช้งานในโปรแกรมหลัก (รูปเดียวกัน) โดยมีการดำเนินการดังนี้

- ให้ค่าโปรแกรมย่อยแก่อะเรย์แต่ละช่อง
- ให้วงรอบ repeat ให้ผู้ใช้ป้อนเลขจำนวนเต็ม 2 จำนวน นำมาลบกันแล้วหาเศษจากการหารด้วย 2 นำไปเป็นอินเด็กซ์ของอะเรย์ ขอให้สังเกตการให้ค่าอาร์กิวเมนต์แก่อะเรย์โปรแกรมย่อยด้วย ได้ผลการรันดังรูปที่ 5

รูปที่ 3 ผลการรันของโปรแกรมตามรูปที่ 2

```
This is ONE
This is TWO
This is TWO
10 is greater than 5 is TRUE
```

รูปที่ 4 แสดงการกำหนดตัวแปรเป็นอะเรย์ของโปรแกรมย่อย และการใช้ตัวโปรแกรมย่อยอยู่ในรูปที่ 1

```
VAR    compArray : ARRAY[-1..1] OF CompareFunct;
       res : Integer;

BEGIN
  compArray[-1] :=  LesserThan;
  compArray[0]  :=  Equal;
  compArray[1]  :=  GreaterThan;

  REPEAT
    Write('Enter 2 integer number ');
    ReadLn(a,b);
    res := (a - b) mod 2;
    WriteLn(compArray[res](a,b));  WriteLn;
  UNTIL  a = 0;

readln;
END.
```

รูปที่ 5 ผลการรันของโปรแกรมตามรูปที่ 4

```
Enter 2 integer number 10 5
10 is greater than 5 is TRUE

Enter 2 integer number 5 10
5 is lesser than 10 is TRUE

Enter 2 integer number 5 5
5 is equal to 5 is TRUE

Enter 2 integer number 0 0
0 is equal to 0 is TRUE
```

โปรแกรมย่อยในเรคอร์ด

ในทำนองเดียวกันเราสามารถจัดให้ตัวแปรแบบโปรแกรมย่อยเป็นฟิลด์หนึ่งของเรคอร์ดได้ดังโปรแกรมในรูปที่ 6 ซึ่งได้กำหนดแบบข้อมูลโปรแกรมย่อย GotoProc แล้วนำไปใช้ในเรคอร์ด

สำหรับการใช้งานก็เช่นกันจะต้องให้ค่าโปรแกรมย่อยแก่ตัวแปรก่อน ส่วนการใช้งานคงเป็นแบบผสมกับเรคอร์ดและใช้กับตัวแปรพอยน์เตอร์ได้เช่นกัน

ผ่านค่าเป็นโปรแกรมย่อย

เราสามารถผ่านค่าเป็นโปรแกรมย่อยหรือตัวแปรโปรแกรมย่อยได้เช่นเดียวกับข้อมูลแบบอื่น และขอยกตัวอย่างเป็นฟังก์ชันอีกเช่นกันเพราะการผ่านค่าในแบบโพรซีเดอร์ได้นำมาใช้จริงในบทความชุดซอฟต์แวร์ชิพแล้ว

ตามรูปที่ 7 เริ่มด้วยการกำหนดแบบข้อมูลฟังก์ชันที่มีอาร์กิวเมนต์เป็นเลขจำนวนเต็มบวก 2 จำนวน และตัวฟังก์ชันเองก็เป็นเลขจำนวนเต็มบวกเช่นกัน มีโปรแกรมย่อยตามแบบข้อมูลที่สองโปรแกรมคือ Add (เพื่อการบวก) และ Mul (เพื่อการคูณ)

โปรแกรมย่อยที่จะรับการผ่านค่าเป็นโปรแกรมย่อยคือ DispResult จะนำค่าจาก frst ถึง fin มาดำเนินการตามโปรแกรมย่อยที่โปรแกรมหลักจะผ่านค่ามาให้แก่ operation เช่น ถ้าเป็นการบวกก็จะบวกค่าตั้งแต่ frst ถึง fin นำผลออกแสดงเป็นลำดับไปดังผลการรันในรูปที่ 8 บรรทัดแรกเป็นผลของการบวก บรรทัดที่สองเป็นผลของการคูณ

กับแอดเดรส

การให้ค่าโปรแกรมย่อยแก่ตัวแปรโปรแกรมย่อยดังกล่าวมา ความจริงแล้วเป็นการให้ค่าแอดเดรสของโปรแกรมย่อยแก่ตัวแปรดังแสดงในรูปที่ ๑ (ตัวโปรแกรม Add และ Mul อยู่ในรูปที่ ๑) ซึ่งได้ให้คำเนินการดังนี้คือ

- ให้ค่าโปรแกรม Add แก่ตัวแปร funcone

- ในเงื่อนไข IF จึงเป็นการเรียกใช้ฟังก์ชัน Add กับฟังก์ชัน Mul การเปรียบเทียบผลการบวกกับการคูณจึงไม่เท่ากัน

- ในเงื่อนไข IF ที่สองจะเป็นการเปรียบเทียบแอดเดรสของ Add กับ Mul จึงได้ผลไม่เท่ากันอีก

- ส่วนใน ELSE IF เป็นการ

รูปที่ ๑ แสดงการใช้โปรแกรมย่อยในเรคอร์ด

```
TYPE
  GotoProc = Procedure(x,y : Byte);
  WindowT = RECORD
    xx1,xx2,yy1,yy2 : Byte;
    curx,cury : Byte;
    SetCursor : GotoProc;
          END;

  WindowPtrT = ^WindowT;

VAR   wnd : WindowT;
      wndPtr : WindowPtrT;

{$F+}
procedure  JumpToXY(x,y : Byte);
begin
  GotoXY(x,y);
end;
{$F-}


BEGIN
  ClrScr;
  wnd.SetCursor := JumpToXY;
  wnd.SetCursor(20,10);  Write('!');
  wndPtr := @wnd;
  wndPtr^.SetCursor(30,10);  Write('!');
readln;
END.
```

รูปที่ ๒ แสดงการผ่านค่าเป็นโปรแกรมย่อย

```
PROGRAM  DemoProcTypeTwo;

TYPE  ArithFunc = Function( frst,scnd : Word ) : Word;

{$F+}
FUNCTION  Add(frst,scnd : Word) : Word;
BEGIN
  Add := frst + scnd;
END;


FUNCTION  Mul(frst,scnd : Word) : Word;
BEGIN
  Mul := frst * scnd;
END;
{$F-}


PROCEDURE  DispResult( operation : ArithFunc; frst,last : Word);
VAR  res,cnt : Word;
BEGIN
  res := frst;  inc(frst);
  for  cnt := frst to last  do
    begin  res := operation(res,cnt);
           Write(res:6)  end;
  WriteLn;
END;


BEGIN
  DispResult(Add,1,10);
  DispResult(Mul,1,10);
END.
```

รูปที่ 8 ผลการรันของโปรแกรมตามรูปที่ 7

```
3     6    10    15    21    28    36    45    55
2     6    24   120   720  5040 40320 35200 24320
```

รูปที่ 9 แสดงการอ้างแอดเดรสของข้อมูลแบบโปรแกรมย่อย ตัวโปรแกรมย่อยอยู่ในรูปที่ 7

```
VAR  funcone,functwo : ArithFunc;
     pntr   : Pointer;
     res    : Word;

BEGIN
  funcone := Add;
  if  funcone(1,2) = Mul(1,2) then  WriteLn('Equal')
  else  WriteLn('Not equal');

  if  @funcone = @Mul  then  WriteLn('funcone is Mul')
  else if @funcone = @Add  then  WriteLn('funcone is Add');

  functwo := Mul;
  pntr := @Add;  res := ArithFunc(pntr)(5,10);  WriteLn(res);
  ArithFunc(pntr) := functwo;  WriteLn(ArithFunc(pntr)(5,10));
  funcone := ArithFunc(pntr);  WriteLn(funcone(5,10));

  pntr := @Add;  @funcone := pntr;  WriteLn(funcone(2,10));
  pntr := @functwo;  WriteLn(ArithFunc(pntr)(2,10));


readln;
END.
```

@@funcone

เทียบแอดเดรสของ Add กับ Add ได้ผลเท่ากัน

- โปรแกรมท่อนต่อไปได้ให้ค่าโปรแกรมย่อย Mul แก่ functwo เพื่อจะนำไปใช้ต่อไป

- ถัดไปเป็นการให้ค่าแอดเดรสของ Add แก่ตัวแปรพอยน์เตอร์ pntr แล้วนำไปใช้ แต่ pntr เป็นพอยน์เตอร์แบบทั่วไป ดังนั้นเมื่อนำไปใช้จึงต้องให้แบบข้อมูลกำกับไว้ด้วย (type cast) ให้ผลของการบวก 5 กับ 10 แก่ res

- อีกแบบหนึ่งของการให้ค่าแอดเดรสของโปรแกรมย่อยในตัวแปรแก่ตัวแปรพอยน์เตอร์คือ ใช้แบบข้อมูลกำกับดังที่ให้ค่าใน functwo แก่ pntr เมื่อให้แสดงผลจึงเป็นผลของการคูณ 5 กับ 10

- ในทำนองเดียวกันสามารถให้ค่าแอดเดรสที่เก็บอยู่ในพอยน์เตอร์ แก่ตัวแปรโปรแกรมย่อยโดยใช้แบบข้อมูลกำกับได้เช่นกันดังที่ให้แก่ funcone ดังนั้นในตอนนี้ funcone จึงแทนโปรแกรมย่อย Mul เมื่อให้แสดงผลจึงเป็นผลของการคูณ 5 กับ 10 อีกเช่นกัน

- อีกแบบหนึ่งของการให้ค่าแอดเดรสแก่ตัวแปรโปรแกรมย่อย คืออ้างถึงแอดเดรสของตัวแปรโปรแกรมย่อยนั้นโดยตรงด้วยการใช้เครื่องหมาย at sign นำหน้าดังที่ใช้กับ funcone ในโปรแกรมท่อนสุดท้าย ซึ่งในกรณีนี้จะทำให้ funcone แทนโปรแกรมย่อย Add เมื่อให้แสดงผลจึงเป็นผลของการบวก 2 กับ 10

ตัวอย่างการใช้งาน

สำหรับตัวอย่างการใช้งานที่เป็นของจริงขอให้ดูในบทความชุดซอฟต์แวร์ชิพไมโครคอมพิวเตอร์ฉบับที่ 60 ครับ